An Algebra of Deterministic Propositional Acceptance Automata (DPAA)

Aurélien Lamercerie, Benoît Caillaud

An Algebra of Deterministic Propositional Acceptance Automata (DPAA)

1st Aurélien Lamercerie
UMR 6074
Univ Rennes, Inria, IRISA
Rennes, France
aurelien.lamercerie@inria.fr

2rd Benoit Caillaud
UMR 6074
Univ Rennes, Inria, IRISA
Rennes, France
benoit.caillaud@inria.fr

***Abstract*—Deterministic Propositional Acceptance Automata (DPAA) are proposed to capture system requirements expressing mandatory and forbidden discrete-time behavior. The main feature of this formalism is that it can express the expected behavior when the system is in a particular state. DPAA are therefore blending together state properties, expressed as propositional formulas, and simple discrete-time temporal properties, expressed as mandatory and forbidden actions whenever a given state property holds. They extend modal transition systems to a propositonal setting, where models are Kripke structures, rather than labelled transition systems. Composition operators on DPAA are provided, making them an Interface Theory, with a refinement relation, parallel composition, conjunction and quotient operators. An implicit representation using characteristic functions is also proposed to limit the time/space computational complexity.**

***Index Terms*—Automata for System Analysis, Interface Theory, Discrete Time Reactive System, Requirements Engineering**

I. Introduction

Contract-based reasoning [1], [2] is a powerful method for software design. Several programming languages [3] offer native support for contracts inspired by Hoare logic [4] and extensions, such as Separation Logic [5]. While finding programming errors early in software development saves considerable time and money, detecting specification errors is even more crucial. Formalizing requirements, analyzing them by computer assisted methods, or monitoring them at runtime, are keys to identify inconsistent, redundant or incomplete requirements.

In the context of reactive system design, Contract Theories [6] offer a flexible formal framework as a generic theory and can support a variety of formalisms and design processes. All these approaches have in common a composition algebra and a concept of refinement, reflecting the decomposition of system-level requirements into several viewpoints and components. We fit to the Contract Theory framework to propose a specification formalism for discrete-time reactive systems, that offers a good compromise in terms of expressiveness, algebraic properties, algorithmic complexity, and ease of use by engineers who are non-computer scientists with any knowledge of temporal logics. The main feature of this theory is that both event and state properties can be taken into account.

Interface Theories aim at providing a merged specification of the expected behavior of a component under design and of the possible environments in which it may run. Typical instances of Interface Theories are *Interface Automata* [7] and *Modal Interfaces* [6], [8], [9]. These automata-theoretic formalisms, inspired by Lynch's Input/Output Automata [10], [11], are capable of capturing both the variability of the possible designs and the uncertainties regarding the possible environments of a component. Component compatibility is also captured by characterizing the environments in which arbitrary realizations of the interfaces may be correctly composed. Interface Theories have also been branded to encompass Moore Machines. These are the Moore or Synchronous Interfaces, presented in the landmark paper [12].

However, they deal only with the Input/Output behavior of reactive components, while, in many cases, it is desirable to relate Input/Output behavior to the component's state.

Our contribution, Deterministic Propositional Acceptance Automata (DPAA), specifically addresses this issue. These automata make it possible to specify discrete, mandatory or forbidden behavior of discrete-time reactive systems. Main characteristic of this formalism is that it allows to express the expected behavior when the system is in a particular state. DPAA combines state properties, expressed as propositional formulas, and behavioral properties, expressed as mandatory or forbidden events. They extend modal transition systems using Kripke structures as models rather than labeled transition systems. It is inspired by the work of Jean-Baptiste Raclet [13] on the Acceptance Sets Specifications and Benveniste et al. [6] on the Contract Theory. As such, it comes with a composition algebra which makes it possible to reason formally on requirements consisting in many elementary specifications.

The explicit manipulation of acceptance sets involves building larger and larger sets with a significant computational complexity. An implicit representation of acceptance sets using characteristic functions is proposed to curb the computational complexity associated with this algebra. Parameters are used for enumerating sets of possibilities, while characteristic functions specify the contents of these sets. This enables a concise representation of acceptance sets.

The paper is organized as follows. Kripke Models (KM) are defined in Section II. These allow to represent the behavior of components and systems. Section III focuses on Acceptance Sets (AS), used to capture with great flexibility the variability

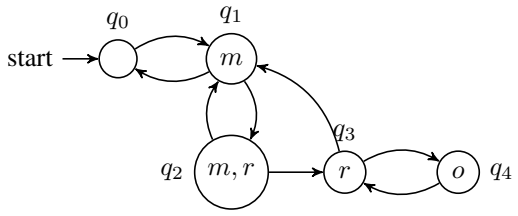


Fig. 1. Modeling of a system representing a shuttle bus

in the implementation of a specification. Implicit representation of AS is also characterized. Finally, as an Automata-theoretic layer above AS, Deterministic Propositional Acceptance Automata (DPAA) and its composition algebra are introduced in Section IV. Those form the main contribution of this paper by exploiting AS through automata.

II. Realisations as Kripke Models (KM)

Systems behavior is defined as non-deterministic Kripke models on a set of atomic propositions. This model allows to express the expected behavior when the system checks state properties, expressed as propositional formulas.

Definition 1: Kripke Model (KM)
A Kripke model on alphabet Π is a tuple $\mathcal{M} = (\Pi, Q, Q_0, \rightarrow, \upsilon)$ such that:

- Π is a set of atomic propositions;
- Q is a set of states;
- $Q_0 \subseteq Q$ is a set of initial states;
- $\rightarrow \subseteq Q \times Q$ is the transition relation between states;
- $\upsilon : Q \mapsto 2^{\Pi}$ is the valuation function.

It is assumed that KMs are stuttering invariant [14], meaning that a KM can loop indefinitely on the same state. Stuttering invariance plays an important role in the parallel composition of KMs. It allows components to evolve asynchronously. Without this assumption, parallel composition would force a strict synchronism between components. Self-loops are implicit.

Example.: A shuttle bus is considered. The focus is on the access doors to the vehicle. The alphabet of atomic propositions is $\Pi = \{m, r, o\}$, where (o) denotes the opening of the doors, (m) the shuttle is in motion, and (r) a stop request is pending. Figure 1 shows a KM modeling for the behaviour of a door. It is assumed the vehicle is initially stopped (q_0). It can move (q_1), and in this case, a stop can still be requested (q_2). The shuttle bus must stop (q_3) in order to have the doors opened (q_4). Only true propositions are noted in states, and self-loops are not represented (they are implicitly in all states).

A trace $u = (u_1, \ldots, u_n)$ is a finite sequence of valuations of the atomic propositions, with $u_i \subseteq \Pi$. Denote $u.v$ the concatenation of traces. The empty trace is denoted ϵ. The transition function extended to words on $(2^{\Pi})^*$ is the function $\Delta : 2^Q \times (2^{\Pi})^* \rightarrow 2^Q$ such that $\forall P \subseteq Q$, $\Delta(P, \epsilon) = P$ and $\forall u \in (2^{\Pi})^*, a \in 2^{\Pi}, \Delta(P, u.a) = \{q' \mid \exists q \in \Delta(P, u)$ such that $q \rightarrow q'$ and $\upsilon(q') = a\}$. This ultimately defines the language of a KM reflecting all possible runs of a KM. The language of a KM $\mathcal{M}$ is defined as $\mathcal{L}(\mathcal{M}) = \{u \mid \Delta(Q_0, u) \neq \emptyset\}$.

The parallel composition, also called product, is an important operation to define the behavior of a system composed of several components. The definition of parallel composition for KMs uses a permutation operator, denoted $\leftrightarrow_{23}$, which gives the tuple $((q_1, q_2), (q_1', q_2'))$ from a tuple $((q_1, q_1'), (q_2, q_2'))$.

Definition 2: Parallel Composition
Let $M_1 = (\Pi_1, Q_1, Q_{0,1}, \rightarrow_1, \upsilon_1)$ and $M_2 = (\Pi_2, Q_2, Q_{0,2}, \rightarrow_2, \upsilon_2)$ be two KMs. The product of M_1 and M_2, denoted $M_1 \times M_2$, is the KM $M = (\Pi, Q, Q_0, \rightarrow, \upsilon)$ such that:

- $\Pi = \Pi_1 \cup \Pi_2$
- $Q = \{(q_1, q_2) \in Q_1 \times Q_2 \mid (\upsilon_1(q_1) \cap \Pi_2) = (\upsilon_2(q_2) \cap \Pi_1)\}$
- $Q_0 = (Q_{0,1} \times Q_{0,2}) \cap Q$
- $\rightarrow = \leftrightarrow_{23} (\rightarrow_1 \times \rightarrow_2) \cap Q^2$
- $\upsilon(q_1, q_2) = \upsilon_1(q_1) \cup \upsilon_2(q_2)$

Note that parallel composition of homogeneous models is a synchronous product, that satisfies the following equalities, modulo renaming: commutativity ($M_1 \times M_2 = M_2 \times M_1$), associativity ($M_1 \times (M_2 \times M_3) = (M_1 \times M_2) \times M_3$) and existence of a neutral element ($M_e = \{\Pi = \emptyset, Q = \{q_0\}, Q_0 = \{q_0\}, \rightarrow, \upsilon\}$, with $\rightarrow = \{(q_0, q_0)\}$ and $\upsilon(q_0) = \emptyset$.

III. Acceptance Sets (AS) to capture variability of realisations

Deterministic Propositional Acceptance Automata (DPAA) defined in next section consists of automata whose states are labelled with particular sets, named Acceptance Sets (AS). We first study the algebraic properties of these sets. We will then be able to extend these properties to DPAA.

Definition 3: Acceptance Sets (AS)
An acceptance set Acc on a set of atomic propositions Π is a set of sets of labels on Π, ie $Acc \in 2^{2^{\Sigma}}$, with $\Sigma = 2^{\Pi}$.

Labels are boolean valuations of a set of atomic propositions Π, and are related to properties of a given system. Each $\alpha \in 2^{\Sigma}$, called "ready set", defines a set of properties that a system must satisfy. An AS Acc is interpreted as a set of possibilities, where a ready set ($\alpha \in Acc$) can be chosen to represent the properties that a system must meet. The variability of realization for a given specification can be captured by an AS, since specifications admit several realizations in general.

Figure 2, proposed further, shows some AS.

A. AS Algebra

An algebra with good properties is necessary for AS to be usable as an interface theory [6]. The algebra consists in a satisfaction relation $\models$, a refinement relation $\preceq$ and composition operators $\wedge$, $\otimes$ and $/$. We start by the definition of an algebra of homogeneous AS, meaning that AS is considered on a uniform alphabet of atomic propositions Π. Then this algebra will be extended to heterogeneous AS, meaning that AS can be considered on different alphabets.

Homogeneous AS Algebra.: The satisfaction relation is the selection of one ready-set among the elements of the AS. Refinement is the inclusion of AS. Conjunction is the intersection of AS, and the product is the pointwise intersection of ready sets. The quotient is defined relatively to the product: it is the right adjoint of the product operator: $X \otimes B \preceq A \Leftrightarrow X \preceq A/B$.

Definition 4: Algebra of homogeneous AS

The following operators compose an algebra for homogeneous AS:

- satisfaction: $\alpha \models Acc$ iff $\alpha \in Acc$;
- refinement: $Acc_1 \preceq Acc_2$ iff $Acc_1 \subseteq Acc_2$;
- conjunction: $Acc_1 \wedge Acc_2 = Acc_1 \cap Acc_2$;
- product: $Acc_1 \otimes Acc_2 = \{\alpha_1 \cap \alpha_2 \mid \alpha_1 \in Acc_1, \alpha_2 \in Acc_2\}$;
- quotient: $Acc_1/Acc_2 = \{\alpha \mid \forall \beta \in Acc_2, \alpha \cap \beta \in Acc_1\}$.

These operators define a complete interface theory. We will now define several operators on labels. Based on these, operators on ready-sets are then defined. These finally make it possible to extend the algebra of homogeneous AS to heterogeneous AS.

Operators on Labels.: Let $\upsilon_1 \subseteq \Pi_1$ and $\upsilon_2 \subseteq \Pi_2$ be two valuations on heterogeneous alphabets Π_1 and Π_2. We note Π_{12} the intersection between these alphabets, ie $\Pi_{12} = \Pi_1 \cap \Pi_2$. Operator $\bowtie$ is a synchronization relation such that $\upsilon_1 \bowtie \upsilon_2$ if, and only if, $\upsilon_1 \cap \Pi_{12} = \upsilon_2 \cap \Pi_{12}$. $\upsilon_1 \sqcup \upsilon_2 \subseteq (\Pi_1 \cup \Pi_2)$ is defined as $\upsilon_1 \sqcup \upsilon_2 = \upsilon_1 \cup \upsilon_2$ if $\upsilon_1 \bowtie \upsilon_2$, undefined otherwise. $\upsilon_1 \sqsubseteq \upsilon_2$ if, and only if $\Pi_1 \supseteq \Pi_2$ and $\upsilon_1 \bowtie \upsilon_2$.

Operators on Ready-sets.: We use $\bowtie$, $\sqcup$ and $\sqsubseteq$ on labels to define $\bowtie$, $\sqcap$, $\sqsubseteq$ and $\downarrow$ on ready sets: $\alpha_1 \bowtie \alpha_2$ if, and only if, $\forall \upsilon_1 \in \alpha_1, \exists \upsilon_2 \in \alpha_2, \upsilon_1 \bowtie \upsilon_2$ and $\forall \upsilon_2 \in \alpha_2, \exists \upsilon_1 \in \alpha_1, \upsilon_2 \bowtie \upsilon_1$; $\alpha_1 \sqcap \alpha_2 = \{\upsilon_1 \sqcup \upsilon_2 \mid \upsilon_1 \in \alpha_1, \upsilon_2 \in \alpha_2, \upsilon_1 \bowtie \upsilon_2\}$; $\alpha_1 \sqsubseteq \alpha_2$ if, and only if, it exists a surjection $f : \alpha_1 \to \alpha_2$ such that $\forall \upsilon_1 \in \alpha_1, \upsilon_1 \sqsubseteq f(\upsilon_1)$. Operator $\downarrow$ specifies a projection of ready-sets on a set of atomic propositions, namely $\alpha_{\downarrow \Pi'} = \{\sigma' : \Pi' \mid \exists \sigma \in \alpha, \sigma \bowtie \sigma'\}$. It is used to hide atomic propositions.

Heterogeneous AS Algebra.: AS algebras are adapted with the operators on ready-sets. In the following definition, we consider Acc on alphabet Π, Acc_1 and Acc_2 on alphabets Π_1 and Π_2 such that $\Pi_2 \subseteq \Pi_1$, and a ready-set $\alpha \subseteq 2^{\Pi_\alpha}$, such that $\Pi \subseteq \Pi_\alpha$.

Definition 5: Algebra of heterogeneous AS

The following operators compose an algebra for heterogeneous AS:

- satisfaction: $\alpha \models Acc$ iff $\alpha_{\downarrow \Pi} \in Acc$;
- refinement: $Acc_1 \preceq Acc_2$ iff $\forall \alpha_1 \in Acc_1, \alpha_1 \models Acc_2$;
- conjunction: $Acc_1 \wedge Acc_2 = \{\alpha_1 \sqcap \alpha_2 \mid \alpha_1 \in Acc_1, \alpha_2 \in Acc_2, \alpha_1 \bowtie \alpha_2\}$;
- product: $Acc_1 \otimes Acc_2 = \{\alpha_1 \sqcap \alpha_2 \mid \alpha_1 \in Acc_1, \alpha_2 \in Acc_2\}$;
- quotient: $Acc_1/Acc_2 = \{\alpha \subseteq 2^{\Pi_1 \cup \Pi_2} \mid \forall \beta \in Acc_2, (\alpha \sqcap \beta)_{\downarrow \Pi_1} \in Acc_1\}$.

Figure 2 illustrates the evaluation of conjunction by applying this method.

Note that we recover the algebra for homogeneous acceptance sets when these definitions are applied to homogeneous alphabets, namely when $\Pi_1 = \Pi_2$.

B. Implicit Representation of AS

The explicit manipulation of AS involves to build larger and larger sets with a crippling computational complexity. For exemple, consider two disjoint alphabets Π_a and Π_b of 10 elements each, with $a \in \Pi_a$ and $b \in \Pi_b$. The union of this two alphabets contains 20 elements. Consider two AS $A = \{\{a\}\}$ and $B = \{\{b\}\}$. The quotient of these two sets results in a very large acceptance set, whose ready sets correspond to all possible behaviors according to the condition that it contains $\sigma = \{ab\}$ and not $\sigma' = \{\overline{a}b\}$, ie $Acc_1/Acc_2 = \{\alpha \subseteq 2^{\Pi_1 \cup \Pi_2} \mid (\exists \sigma \subseteq \Pi_1 \cup \Pi_2 \text{ s.t. } ab \bowtie \sigma \text{ and } \sigma \in \alpha) \text{ and } (\forall \sigma : \Pi_1 \cup \Pi_2, \overline{a}b \bowtie \sigma \Rightarrow \sigma \notin \alpha)\}$. The size of this set is approximately $2^{2^{18}}$.

Implicit Representation.: This justifies seeking an implicit representation, using characteristic functions representing ready-sets. Since we also would like to avoid enumerating the ready sets that belong to an AS, we use a set of parameters for enumerating ready sets. Labels on the set of atomic propositions Π are defined as boolean functions $\overrightarrow{\pi} : \Pi \to \mathbb{B}$. In the same way, parameter values are encoded, using a set of boolean variables $\mathcal{P} = \{p_1, ..., p_m\}$. Thus, parameters values are boolean functions $\overrightarrow{p} : \mathcal{P} \to \mathbb{B}$.

Definition 6: Implicit representation of AS

The implicit representation of an acceptance set Acc is a tuple $(\Pi, \mathcal{P}, \psi, \phi)$ such that:

- Π is an alphabet of atomic propositions,
- $\mathcal{P}$ is a set of parameter variables,
- $\psi : (\mathcal{P} \to \mathbb{B}) \to \mathbb{B}$ is the characteristic function for parameter values,
- $\phi : (\mathcal{P} \to \mathbb{B}) \to (\Pi \to \mathbb{B}) \to \mathbb{B}$ is the characteristic function for ready sets.

Intuitively, the function ψ is used to specify which parameter values are meaningful, while the function ϕ is used to define the contents of a ready set given parameter values. For example, acceptance set $Acc = \{\{a\}, \{a, ab\}, \{a, ac\}\}$ on alphabet $\Pi = \{a, b, c\}$ can be implicitly defined with two parameters ($\mathcal{P} = \{p, q\}$), and the characteristic functions $\psi = \neg p \vee \neg q$ and $\phi = \sigma_1 \vee (p \Rightarrow \sigma_2) \vee (q \Rightarrow \sigma_3)$. Note that these characteristic functions can be encoded as boolean functions, and represented as a Binary Decision Diagram (BDD) [15].

Algebraic Properties.: Recall that $\overrightarrow{\pi} : \Pi \to \mathbb{B}$ denotes a label on Π, and $\overrightarrow{p} : \mathcal{P} \to \mathbb{B}$ denotes valuation of parameter variables $\mathcal{P}$. The following theorem defines the expected algebraic operations, whose consistency with the definitions presented in the introduction can be checked.

Theorem 1: Algebraic Properties of AS implicit representation

Let Acc_1 and Acc_2 be two acceptance sets such that $Acc_i = (\Pi_i, \mathcal{P}_i, \psi_i, \phi_i)$. The following operations define an algebra for AS implicit representation:

- satisfaction: $\alpha \models Acc$ if, and only if, $\exists \overrightarrow{p}, [\psi(\overrightarrow{p}) \wedge \forall \overrightarrow{\pi}, \phi(\overrightarrow{p}, \overrightarrow{\pi}) \Leftrightarrow \alpha(\overrightarrow{\pi})]$.

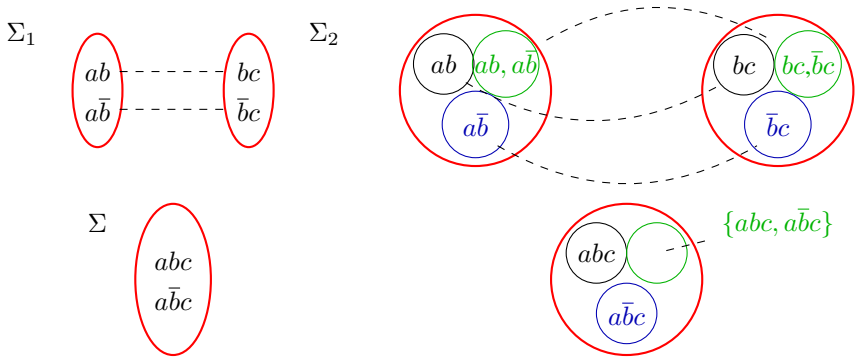


Fig. 2. Example of conjunction (synchronization of labels on the left, synchronization of ready sets on the right, result down)

- refinement: $Acc_1 \preceq Acc_2$ if, and only if
  1) $\Pi_1 \supseteq \Pi_2$,
  2) $\forall \overrightarrow{p_1}, \psi_1(\overrightarrow{p_1}) \Rightarrow [\exists \overrightarrow{p_2}, (\psi_2(\overrightarrow{p_2}) \ \wedge \ \forall \overrightarrow{\pi_1}, (\phi_1(\overrightarrow{p_1}, \overrightarrow{\pi_1}) \Rightarrow \phi_2(\overrightarrow{p_2}, \overrightarrow{\pi_1} \downarrow_{\mathcal{P}_2})))]$.
- conjunction: $Acc_1 \wedge Acc_2 = (\Pi, \mathcal{P}, \psi, \phi)$ such that:
  - $\Pi = \Pi_1 \cup \Pi_2$,
  - $\mathcal{P} = \mathcal{P}_1 \times \mathcal{P}_2 \times Q$, with $Q = 2^\Sigma$ and $\Sigma = 2^\Pi$,
  - $\psi(p_1, p_2, q) = \psi_1(p_1) \ \wedge \ \psi_2(p_2) \ \wedge \ \Delta(p_1, p_2) \ \wedge \ \Gamma_{p_1,p_2}(q)$, with:
    - $\Delta(p_1, p_2) = \forall \sigma \in \Sigma, (\phi_1(p_1, \sigma) \Leftrightarrow \phi_2(p_2, \sigma))$,
    - $\Gamma_{p_1,p_2}(q) = [\forall \sigma \in q, \phi_1(p_1, \sigma_{\downarrow_{\Sigma_1}}) \ \wedge \ \phi_2(p_2, \sigma_{\downarrow_{\Sigma_2}})] \wedge$
      $[\forall \sigma_1 \in \Sigma_1, \phi_1(p_1, \sigma_1) \Rightarrow \exists \sigma \in q, \sigma_{\downarrow_{\Sigma_1}} = \sigma_1] \wedge$
      $[\forall \sigma_2 \in \Sigma_2, \phi_2(p_2, \sigma_2) \Rightarrow \exists \sigma \in q, \sigma_{\downarrow_{\Sigma_2}} = \sigma_2]$,
  - $\phi(p_1, p_2, q)(\sigma) = (\sigma \in q)$.
- product: $Acc_1 \otimes Acc_2 = (\Pi, \mathcal{P}, \psi, \phi)$ such that:
  - $\Pi = \Pi_1 \cup \Pi_2$,
  - $\mathcal{P} = \mathcal{P}_1 \times \mathcal{P}_2$,
  - $\psi(p_1, p_2) = \psi_1(p_1) \ \wedge \ \psi_2(p_2)$,
  - $\phi(p_1, p_2)(\sigma) = \phi_1(p_1, \sigma_{\downarrow_{\Pi_1}}) \ \wedge \ \phi_2(p_2, \sigma_{\downarrow_{\Pi_2}})$.
- quotient: $Acc_1/Acc_2 = (\Pi, \mathcal{P}, \psi, \phi)$ such that:
  - $\Pi = \Pi_1 \cup \Pi_2$,
  - $\mathcal{P} = \mathcal{P}_1 \times Q$, with $Q = 2^\Sigma$ and $\Sigma = 2^\Pi$,
  - $\psi(p_1, p_2) = \psi_1(p_1) \ \wedge \ \forall \sigma \in q, \neg \Delta_{P_2}(q)$, with:
    - $\Delta_{P_2}(q) = \exists p_2 \in \mathcal{P}_2, (\psi_2(p_2) \wedge \phi_2(p_2)(\sigma))$,
  - $\phi(p_1, q)(\sigma) = \phi_1(p_1)(\sigma_{\downarrow_{\Sigma_1}}) \ \wedge \ \Gamma(q)(\sigma)$, with $\Gamma(q)(\sigma) = q \in \sigma$.

Proof 1: We can check for each of these operations the calculated sets are consistent with the definitions presented in the introduction.

For conjunction and quotient, set Q is introduced in a lazy way. The principle is to calculate the synchronization of labels to construct the part graph for labels on the new alphabet. We deduce the part graph for ready sets. In the worst case, the set Q can indeed be very large ($2^{\mathcal{P}}$). In practice, the size of this set is much more reasonable.

This representation reduces the size of defined sets. Consider the example presented at the beginning of the section, with AS $A = \{\{a\}\}$ and $B = \{\{b\}\}$. We have seen that the classical representation of their quotient resulted in a set whose dimension is very important. It can be implicitly defined with only two parameters p and q, and the characteristic functions $\psi = p \vee \neg q$ and $\phi = (p \Rightarrow \{ab\}) \wedge (q \Rightarrow \{\overline{a}b\})$.

IV. Specifications as Deterministic Propositional Acceptance Automata (DPAA)

Deterministic Propositional Acceptance Automata (DPAA) are used to capture state properties, expressed as propositional formulas, and variety of implementations, expressed as acceptance sets. Starting from its initial state, an acceptance automaton observes the runs of a KM, and approves it if, for each KM state, its set of successor states matches with one of the acceptance sets of the specification.

A. Syntax and Semantics of DPAA

An AS is interpreted as a set of valuations. Each state of a given automaton can be associated with several AS. These AS correspond intuitively to sets of possible successors of an observed state of a KM.

Definition 7: Deterministic Propositional Acceptance Automaton (DPAA)
A Deterministic Propositional Acceptance Automaton on the alphabet Π, abbreviated by DPAA, is a tuple $\mathcal{A} = (\Pi, R, r_0, \Rightarrow, \varphi, Acc)$ such that Π is a set of atomic propositions, R a set of states, r_0 the initial state, and:

- $\Rightarrow \subseteq R \cup \{r_0\} \times R$ is the transition relation between states;
- $\varphi : R \rightarrow \mathbf{Prop}(\Pi)$ is a valuation function associating a logical formula to each state;
- $\forall r \in R, \forall r_1, r_2 \in R, r \Rightarrow r_1$ and $r \Rightarrow r_2$ and $\varphi(r_1) \wedge \varphi(r_2)$ satisfiable implies $r_1 = r_2$;
- $Acc : R \rightarrow 2^{2^{2^\Pi}}$ is a mapping associating an acceptance set to each state.

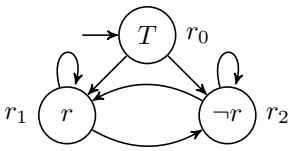
A DPAA is said reduced if, and only if:

1) $\forall r \in R, Acc(r) \neq \emptyset$,
2) $\forall r \in R, \forall \alpha \in Acc(r), \forall \sigma \in \alpha, \exists r' \in R, r \Rightarrow r'$ and $\sigma \models \varphi(r')$.

Note that it is not required that $r_0 \in R$. In particular, when $r_0 \notin R$, then the automaton need not be deterministic in its initial state. Furthermore, the initial state may have the empty

set as acceptance set in a reduced DPAA. Likewise, stuttering invariance is not required for DPAA.

Example.: Consider the shuttle bus on figure 1 with atomic propositions $\Pi = \{m, r, o\}$, and the properties governing the opening of access doors to the shuttle. Figure 3 shows a DPAA representing the following requirement: "the shuttle bus doors opens upon stop request of a passenger". The AS associated with each state are defined in a table.



	$Acc(r_i)$
r_0	$2^{2^{2^\Pi}}$
r_1	$\{\alpha \mid \forall \sigma \in \alpha, o \in \sigma\}$
r_2	$2^{2^{2^\Pi}}$

Fig. 3. DPAA for requirements R_1 and R_2.

Satisfaction.: The satisfaction relation between KMs and DPAA formalizes whether a KM conforms to a DPAA. In other words, a DPAA allowing to express some requirements to be respected by the model, this relation allows us to know if the KM respects these requirements, or on the contrary violates it. To decide on the existence of a satisfaction relation between a KM and a DPAA, we consider a simulation relation between these two formalisms.

Definition 8: Simulation from a KM to a DPAA

The simulation relation between a KM $\mathcal{M}$ and a DPAA $\mathcal{A}$, denoted $\varrho(\mathcal{M}, \mathcal{A})$, is the least relation $\varrho \subseteq Q \times R$ such that:

1) $\forall q_0 \in Q_0, (q_0, r_0) \in \varrho$;
2) $\forall (q, r) \in \varrho$, $\forall q' \in Q, \forall r' \in R$, $q \rightarrow q'$ and $r \Rightarrow r'$ and $\upsilon(q') \models \varphi(r')$ implies $(q', r') \in \varrho$.

Note that conditions of definition 8 give an inductive construction of ϱ. Therefore, there exists a unique simulation ϱ from a model $\mathcal{M}$ to a DPAA $\mathcal{A}$.

We note $\Pi_\mathcal{M}$ the alphabet for a KM $\mathcal{M}$, $\Pi_\mathcal{A}$ the alphabet for a DPAA $\mathcal{A}$. We introduce the operator $\downarrow_{\Pi'}$, which denotes the reduction of a valuation σ to the part relating to the alphabet Π', for a valuation based on an alphabet Π such that $\Pi' \subseteq \Pi$.

Definition 9: Consistent Simulation $\varrho(\mathcal{M}, \mathcal{A})$

Consider $\Pi_\mathcal{M}$ and $\Pi_\mathcal{A}$ such that $\Pi_\mathcal{A} \subseteq \Pi_\mathcal{M}$. The simulation $\varrho(\mathcal{M}, \mathcal{A})$ is consistent if, and only if, $\forall (q, r) \in \varrho(\mathcal{M}, \mathcal{A}), \exists \alpha \in Acc(r)$ such that:

1) $\forall q' \in Q, q \rightarrow q'$ implies $(\upsilon(q') \downarrow_{\Pi_\mathcal{A}}) \in \alpha$,
2) $\forall \sigma \in \alpha, \exists q' \in Q$ such that $q \rightarrow q'$ and $\sigma = (\upsilon(q') \downarrow_{\Pi_\mathcal{A}})$.

Definition 10 allows to have a procedure to decide whether a KM satisfies, or not, a DPAA without having to calculate the associated support languages. This procedure consists in calculating the simulation relation, the complexity of which is quadratic [16].

Definition 10: A KM $\mathcal{M}$ is a model of a DPAA $\mathcal{A}$ if, and only if, the simulation relation $\varrho(\mathcal{M}, \mathcal{A})$ is consistent.

B. DPAA Algebra

DPAA are proposed as a specification formalism capable of capturing both Input/Output behavior and state properties of a component. This formalism comes with a composition algebra, completed by the following relation and operators:

Refinement relation $\preceq$, such that $A_1 \preceq A_2$ if, and only if, for all M, $M \models A_1$ implies $M \models A_2$

Product operator, $A_1 \otimes A_2 = \min\{A \text{ such that for all } M_1, M_2, M_1 \models A_1 \text{ and } M_2 \models A_2 \text{ implies } M_1 \times M_2 \models A\}$

Conjunction operator, $A_1 \wedge A_2 = \max\{A \text{ such that } A \preceq A_1 \text{ and } A \preceq A_2\}$

Quotient operator, $A_1/A_2 = \max\{A \text{ such that } A \otimes A_2 \preceq A_1\}$.

Refinement.: The classical approach for the development of complex systems involves the proposal of a global specification divided into several more precise specifications relating to the various points of the system. This approach is called refinement. One of our goals is to ensure consistency between these different points of view, i.e. to have a formalism that contains a refinement operation.

Definition 11: $\mathcal{A}_1$ is a refinement of $\mathcal{A}_2$, noted $\mathcal{A}_1 \preceq \mathcal{A}_2$, if, and only if, for all models $\mathcal{M}$, $\mathcal{M} \models \mathcal{A}_1$ implies $\mathcal{M} \models \mathcal{A}_2$.

A simulation relation is defined between two DPAA. This is useful to determine if a given automaton $\mathcal{A}_1$ is a refinement of an other automaton $\mathcal{A}_2$, without having to consider all possible models. The important point of this relation implies to consider the observation scope of each pair of states, r_1 of $\mathcal{A}_1$ and r_2 of $\mathcal{A}_2$. A state r_1 of $\mathcal{A}_1$ is linked to a state r_2 of $\mathcal{A}_2$ if the observation scope of r_1 intercepts the observation scope of r_2, in the sense that a run of model (or word) observed by r_1 would also be observed by r_2.

Definition 12: The simulation relation between two DPAA $\mathcal{A}_1$ and $\mathcal{A}_2$, noted $\eta(\mathcal{A}_1, \mathcal{A}_2)$, is the least relation $\eta \subseteq \mathcal{R}_1 \times \mathcal{R}_2$ such that:

1) $(r_{1,0}, r_{2,0}) \in \eta$
2) $\forall (r_1, r_2) \in \eta$, $\forall r_1' \in R_1, \forall r_2' \in \mathcal{R}_2$, $(r_1 \Rightarrow_1 r_1')$ and $(r_2 \Rightarrow_2 r_2')$ and $(\exists \alpha \in Acc_1(r_1), \exists \sigma \in \alpha, \sigma \models \varphi_1(r_1') \wedge \varphi_2(r_2'))$ implies $(r_1', r_2') \in \eta$

Definition 13 formalizes the notion of consistent simulation, necessary to demonstrate refinement.

Definition 13: The simulation $\eta(\mathcal{A}_1, \mathcal{A}_2)$ is said to be consistent if, and only if, the following axiom holds:

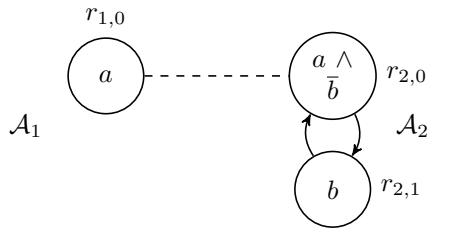
- $\forall (r_1, r_2) \in \eta, Acc(r_1) \subseteq Acc(r_2)$.

Figure 4 shows a representation of a simulation relation (dashed) between two DPAA $\mathcal{A}_1$ (on the left) and $\mathcal{A}_2$ (on the right), and a table giving the AS associated with the states for each of these automata.

It is finally possible to propose a theorem usable to decide the refinement between two DPAA, based on the simulation relation which is directly computable.

Theorem 2: Theorem for Refinement

$\mathcal{A}_1 \preceq \mathcal{A}_2$ if, and only if, the simulation $\eta(\mathcal{A}_1, \mathcal{A}_2)$ is consistent (definition 12).



	$Acc(\mathcal{A}_1)$	$Acc(\mathcal{A}_2)$
$r_{1,0}$	$\{a\overline{b}\}$	
$r_{2,0}$		$\{a\overline{b}, ab\}, \{\overline{a}\overline{b}\}$
$r_{2,1}$		$\{\overline{a}b\}$

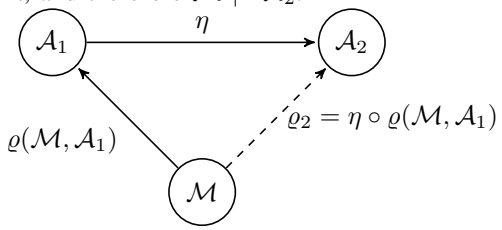
Fig. 4. Simulation relation between $\mathcal{A}_1$ (on the left) and $\mathcal{A}_2$ (on the right)

Proof 2:

Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be two DPAA. Let $\eta(\mathcal{A}_1, \mathcal{A}_2)$ the simulation of $\mathcal{A}_1$ to $\mathcal{A}_2$ satisfying the conditions of definition 12. Consider the simulation relations $\varrho(\mathcal{M}, \mathcal{A}_1)$ and $\varrho(\mathcal{M}, \mathcal{A}_2)$ between any model $\mathcal{M}$ and $\mathcal{A}_1$, $\mathcal{A}_2$ respectively.

Implication 1: $\eta(\mathcal{A}_1, \mathcal{A}_2)$ consistent implies $\mathcal{A}_1 \preceq \mathcal{A}_2$

Let $\mathcal{M}$ be a model such that $\mathcal{M} \models \mathcal{A}_1$. The simulation relation $\varrho(\mathcal{M}, \mathcal{A}_1)$ is consistent (Definition 9). Consider the relation $\varrho_2 = \eta(\mathcal{A}_1, \mathcal{A}_2) \circ \varrho(\mathcal{M}, \mathcal{A}_1)$. We have $\varrho_2 \subseteq Q \times R_2$ by construction. We shall prove that $\varrho(\mathcal{M}, \mathcal{A}_2) \subseteq \varrho_2$ and that ϱ_2 is consistent. This would prove that $\varrho(\mathcal{M}, \mathcal{A}_2)$ is also consistent, and therefore $\mathcal{M} \models \mathcal{A}_2$.



Consider the support sets $\mathcal{R}_n$ of $\mathcal{A}_2$ such that :

1) $\mathcal{R}_0 = \{r_{2,0}\}$,
2) $\mathcal{R}_{n+1} = \{r' \mid r \Rightarrow_2 r' \wedge r \in \mathcal{R}_n\} \cup \mathcal{R}_n$.

Remark that, by construction, $\forall n, \mathcal{R}_n \subseteq R_2$ and there exists m such that $\mathcal{R}_m$ contains all reachable states of $\mathcal{A}_2$ (and therefore, $\forall n > m, \mathcal{R}_n = \mathcal{R}_m$). So, consider the following property P_n: $\forall r_2 \in \mathcal{R}_n, (q, r_2) \in \varrho(\mathcal{M}, \mathcal{A}_2)$ implies $(q, r_2) \in \varrho_2$. We prove P_n is true for all n, and therefore $\varrho(\mathcal{M}, \mathcal{A}_2) \subseteq \varrho_2$:

1) According to definition 8, condition 1, $\forall q_0 \in Q_0$, we have $(q_0, r_{1,0}) \in \varrho(\mathcal{M}, \mathcal{A}_1)$ and $(q_0, r_{2,0}) \in \varrho(\mathcal{M}, \mathcal{A}_2)$. According to condition 1 of definition 12, we have $(r_1, r_2) \in \eta(\mathcal{A}_1, \mathcal{A}_2)$ by construction, and $(q_0, r_{2,0}) \in \varrho_2$ because $(r_1, r_2) \circ (q_0, r_1) = (q_0, r_2)$, which proves that P_0 holds.

2) Assume P_n holds: $\forall r_2 \in \mathcal{R}_n$, we have $(q, r_2) \in \varrho(\mathcal{M}, \mathcal{A}_2)$ implies $(q, r_2) \in \varrho_2$. According to condition 2 of definition 8, $\forall (q, r_2) \in \varrho(\mathcal{M}, \mathcal{A}_2)$, $\forall q' \in Q$, $\forall r_2' \in R_2$ such that $q \rightarrow q'$ and $r_2 \Rightarrow_2 r_2'$, we have $(q', r_2') \in \varrho(\mathcal{M}, \mathcal{A}_2)$ if $\upsilon(q') \models \varphi(r_2')$. According to the assumption that $\mathcal{M}$ is model of $\mathcal{A}_1$, $\varrho(\mathcal{M}, \mathcal{A}_1)$ is consistent and $\exists \alpha \in Acc(r_1), \exists \sigma \in \alpha, \upsilon(q') \models \sigma$ (for $q' \in Q, q \rightarrow q'$). According to the assumption that $\mathcal{A}_1$ is consistent, there is necessarily a state $r_1' \in R_1$ such that $r_1 \Rightarrow_1 r_1'$ and $\exists \alpha \in Acc(r_1), \exists \sigma \in \alpha, \upsilon(q') \models \sigma$, therefore $\upsilon(q') \models \varphi(r_1')$. According to condition 2 of definition 12, we have $(r_1', r_2') \in \eta(\mathcal{A}_1, \mathcal{A}_2)$ by construction, and $(q', r_2') \in \varrho_2$ because $(r_1', r_2') \circ (q', r_1') = (q', r_2')$. We verify that $\forall r_2' \in \mathcal{R}_{n+1}, (q', r_2') \in \varrho(\mathcal{M}, \mathcal{A}_2)$ implies ϱ_2, therefore proving that P_{n+1} holds.

Hence we conclude that P_n holds for all n and that $\varrho(\mathcal{M}, \mathcal{A}_2) \subseteq \varrho_2$.

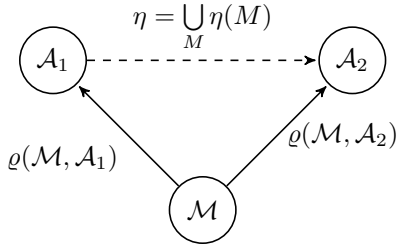
We shall now prove that ϱ_2 is consistent:

- We have $\forall (r_1, r_2) \in \eta, Acc(r_1) \subseteq Acc(r_2)$ (definition 13). $\forall (q, r_1) \in \varrho$, it exists $\alpha \in Acc(r_1)$ verifying conditions of definition 8. By construction, for all $(q, r_2) \in \varrho_2$, there exists r_1 such that $(r_1, r_2) \in \eta(\mathcal{A}_1, \mathcal{A}_2)$ and $(q, r_1) \in \varrho(\mathcal{M}, \mathcal{A}_1)$. According to definition 13, this ensures that it exists $\alpha \in Acc(r_2)$ verifying conditions of definition 8 for all $(q, r_2) \in \varrho$.

So, we conclude that $\varrho(\mathcal{M}, \mathcal{A}_2)$ is also consistent, and therefore $\mathcal{M} \models \mathcal{A}_2$.

Implication 2: $\mathcal{A}_1 \preceq \mathcal{A}_2$ implies $\eta(\mathcal{A}_1, \mathcal{A}_2)$ consistent

Given $\mathcal{M}$, consider $\eta(\mathcal{M}) = \varrho(\mathcal{M}, \mathcal{A}_2) \circ \varrho(\mathcal{M}, \mathcal{A}_1)^{-1}$. Recall that $\varrho(\mathcal{M}, \mathcal{A}_2) \subseteq Q \times R_2$, $\varrho(\mathcal{M}, \mathcal{A}_1) \subseteq Q \times R_1$. Therefore, we have $\varrho(\mathcal{M}, \mathcal{A}_1)^{-1} \subseteq R_1 \times Q$ and $\eta(\mathcal{M}) \subseteq R_1 \times R_2$. Define η as the union of all $\eta(M)$: $\eta = \bigcup_{\mathcal{M}} \eta(\mathcal{M})$. We shall prove that $\eta(\mathcal{A}_1, \mathcal{A}_2) \subseteq \eta$ and that η is consistent. This would prove that $\eta(\mathcal{A}_1, \mathcal{A}_2)$ is also consistent.



Consider the support sets $\mathcal{R}_{1,n}$ of $\mathcal{A}_1$ and $\mathcal{R}_{2,n}$ of $\mathcal{A}_2$ such that, for $i = 1, 2$:

1) $\mathcal{R}_{i,0} = \{r_{i,0}\}$,
2) $\mathcal{R}_{i,n+1} = \{r' \mid r \Rightarrow_i r' \wedge r \in \mathcal{R}_{i,n}\} \cup \mathcal{R}_{i,n}$.

We can remark that, by construction, $\forall n, \mathcal{R}_{i,n} \subseteq R_i$ and it exists m such that $\mathcal{R}_{i,m}$ contains all accessible states of $\mathcal{A}_i$ (and therefore, $\forall n > m, \mathcal{R}_{i,n} = \mathcal{R}_{i,m}$). Moreover, it is possible to match any set R_n with a reference model $\mathcal{M}$ such that $\mathcal{M} \models \mathcal{A}_1$: this model is built by extension starting from

the initial state, and by choosing an acceptance set α each time, each $\sigma \in \alpha$ being associated with a new state of the model.

So, consider the following property P_n: $\forall r_1 \in \mathcal{R}_{1,n}, r_2 \in \mathcal{R}_{2,n}, (r_1, r_2) \in \varrho(\mathcal{A}_1, \mathcal{A}_2)$ implies $(r_1, r_2) \in \eta$. We prove P_n is true for all n, and therefore $\eta(\mathcal{A}_1, \mathcal{A}_2) \subseteq \eta$:

1) Consider any model, Q_0 be the set of initial states for these models. According to condition 1 of definition 8, we have $(q_0, r_{1,0}) \in \varrho(\mathcal{M}, \mathcal{A}_1)$ and $(q_0, r_{2,0}) \in \varrho(\mathcal{M}, \mathcal{A}_2)$ for all $q_0 \in Q_0$. Relation η being defined as the union of the simulation relation for all models, we have $(r_{1,0}, r_{2,0}) \in \eta$ by construction.

2) Assume P_n holds: $\forall r_1 \in \mathcal{R}_{1,n}, r_2 \in \mathcal{R}_{2,n}, (r_1, r_2) \in \varrho(\mathcal{A}_1, \mathcal{A}_2)$ implies $(r_1, r_2) \in \eta$. Consider any $(r_1, r_2) \in \eta(\mathcal{A}_1, \mathcal{A}_2) \cap (\mathcal{R}_{1,n} \times \mathcal{R}_{2,n})$. According to condition 2 of definition 12, $\forall r_1' \in R_1$, $\forall r_2' \in R_2$ such that $r_1 \Rightarrow_1 r_1'$, $r_2 \rightarrow r_2'$, $(r_1', r_2') \in \eta(\mathcal{A}_1, \mathcal{A}_2)$ implies that it exists a set $\alpha \in Acc_1(r_1)$ and a valuation $\sigma \in \alpha$ satisfying $(\varphi(r_1') \wedge (\varphi(r_2'))$. Starting from the reference model $\mathcal{M}$ of R_n, it is possible to construct by extension a model $\mathcal{M}'$ such that $\exists q \in Q, readyset_{\mathcal{M}'}(q) = \alpha$. In this case, we have $(r_1', r_2') \in \eta(\mathcal{M}')$. η is defined as the union of the simulation relation for all models, which included this model. According to condition 2 of definition 8, we have $(r_1', r_2') \in \eta$ by construction.

We now prove that η is consistent. Consider $(r_1, r_2) \in \eta$. Suppose that $Acc(r_1) \nsubseteq Acc(r_2)$. This would mean that there exists σ such that $\sigma \in Acc(r_1)$ and $\sigma \notin Acc(r_2)$. From r_0, construct by extension a model $\mathcal{M}$ by choice for all state q of these model an acceptance set as $Ready_{\mathcal{M}}(q)$, with a state q_1 such that $Ready_{\mathcal{M}(q_1)} = \sigma$. Our initial hypothesis implies that $M \models A_2$, and therefore $\sigma \in \alpha(r_2)$ (axiom 2.a of the definition 8). We deduce that $\sigma \in \alpha(r_1)$ and $\sigma \in \alpha(r_2)$, which implies that $\alpha(r_1) \subseteq \alpha(r2)$.

Thus, we conclude the existence of a simulation satisfying the conditions of the theorem.

Composition operators are an adaptation of operators defined on models such as modal interfaces. Alphabet, automaton structure and state formulas are obtained in a similar way to the conjunction, product and quotient: the alphabet by union, the structure by cartesian product and the formulas by conjunction. The states of the models observed are the same for these three operations. The difference are the acceptance sets. These are obtained by applying the corresponding operations of AS algebra, with a small additional adaptation for quotient. The consistency of these theorems with the axioms of an Interface Theory holds by construction. Some details are given below.

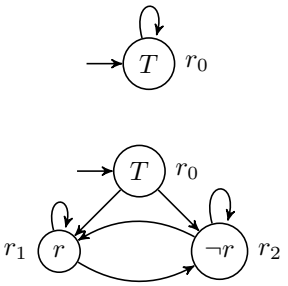
Conjunction: The requirements are to specify the expected properties of the system being designed. They are a way by which original manufacturers interacts with suppliers. and the interpretation of a requirements document is the conjunction of all these implications. This same concept should also be valid for defining the combination of different viewpoints of requirements such as function, safety or energy, based on different modeling frameworks that interact.

Theorem 3: Conjunction $\mathcal{A}_1 \wedge \mathcal{A}_2$ is the DPAA $\mathcal{A} = (\Pi, R, r_0, \Rightarrow, \varphi, Acc)$ such that:

- $\Pi = \Pi_2 \cup \Pi_1$;
- $R = R_1 \times R_2$;
- $r_0 = (r_{1,0}, r_{2,0})$;
- $(r_1, r_2) \Rightarrow (r_1', r_2')$ if, and only if, $r_1 \Rightarrow_1 r_1'$ and $r_2 \Rightarrow_2 r_2'$;
- $\varphi(r_1, r_2) = \varphi_1(r_1) \ \wedge \ \varphi_2(r_2)$;
- $Acc(r_1, r_2) = Acc_1(r_1) \wedge Acc_2(r_2)$.

Example.: Figure 6 shows the conjunction of DPAAs representing two requirements about shuttle bus example (Fig. 5):

- ($\mathcal{R}_1$) "the shuttle bus doors can not open when the shuttle bus is in motion",
- ($\mathcal{R}_2$) "the shuttle bus doors opens on request of a passenger wanting to get off or to get on the shuttle bus".



	$Acc(r_i)$
r_0	$\{\alpha \mid \forall \sigma \in \alpha, m \in \sigma \Rightarrow o \notin \sigma\}$

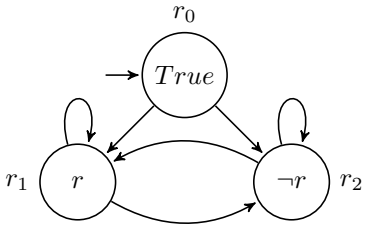
	$Acc(r_i)$
r_0	$2^{2^{2^{\Pi}}}$
r_1	$\{\alpha \mid \forall \sigma \in \alpha, o \in \sigma\}$
r_2	$2^{2^{2^{\Pi}}}$

Fig. 5. DPAA for requirement R_1 and R_2.

Product: Each supplier may have to work on a specific requirements document. For each of these subsystems, a contract can be defined. Integration of different components results in the composition of the subsystems defining the main architecture, reflected by the parallel composition of contracts.

Theorem 4: Parallel composition $\mathcal{A}_1 \otimes \mathcal{A}_2$ is the DPAA $\mathcal{A} = (\Pi, R, r_0, \Rightarrow, \varphi, Acc)$ such that:

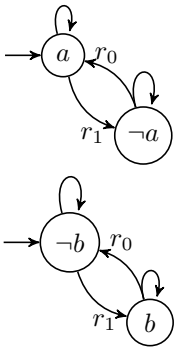
- $R = R_1 \times R_2$;
- $r_0 = (r_{1,0}, r_{2,0})$;
- $(r_1, r_2) \Rightarrow (r_1', r_2')$ if, and only if, $r_1 \Rightarrow_1 r_1'$ and $r_2 \Rightarrow_2 r_2'$;
- $\varphi(r_1, r_2) = \varphi_1(r_1) \ \wedge \ \varphi_2(r_2)$;
- $Acc(r_1, r_2) = Acc_1(r_1) \otimes Acc_2(r_2)$.



	$Acc(r_i)$
r_0	$\{\alpha \mid \alpha \models \Box(m \Rightarrow \neg o)\}$
r_1	$\{\alpha \mid \alpha \models \Box o \wedge (m \Rightarrow \neg o)\} = \{\alpha \mid \alpha \models \Box(o \wedge \neg m)\}$
r_2	$\{\alpha \mid \alpha \models \Box(m \Rightarrow \neg o)\}$

Fig. 6. Conjunction of two DPAA

Example.: We illustrate this operator considering DPAA $\mathcal{A}_1$ and $\mathcal{A}_2$ such that $\mathcal{A}_1$ expresses that b is possible only after a, while $\mathcal{A}_2$ expresses that b is necessarily followed by a (Fig. 7). Figure 8 corresponds to application of product on DPAAs $\mathcal{A}_1$ and $\mathcal{A}_2$.



	$Acc(r_i)$
r_0	$\{a\}, \{a, ab\}, \{a, ab, b\}, \{ab\}, \{ab, b\}, \{b\}, \emptyset$
r_1	$\{a\}, \emptyset$

	$Acc(r_i)$
r_0	$2^{2^{2^{\Pi}}}$
r_1	$\{a\}, \{a, ab\}, \{ab\}$

Fig. 7. $\mathcal{A}_1$ expresses that b is possible only after a, while $\mathcal{A}_2$ expresses that b is necessarily followed by a.

Quotient: The quotient operation is the right-adjoint of the product operator. It is characterized by the following equation:

$$\mathcal{A}_1/\mathcal{A}_2 = max\{A \mid A2 \times A \leq A\}.$$

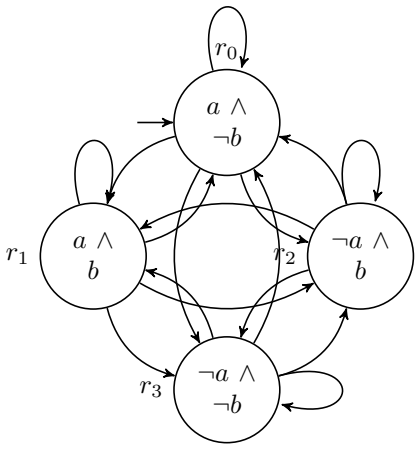
It can be used in a compositional reasoning approach, to decompose a system-wide requirement into component-level specifications (using the same method as in Chapter 10 of [6]) or whenever component reuse is sought: $\mathcal{A}_1/\mathcal{A}_2$ characterizes the realizations $\mathcal{M}$ such that any realization of $\mathcal{A}_2$ composed with $\mathcal{M}$ is a realization of $\mathcal{A}_1$. The quotient of two DPAA is computed as follows:

Theorem 5: Quotient $\mathcal{A}_1/\mathcal{A}_2$ is the DPAA $\mathcal{A} = (\Pi, R, r_0, \Rightarrow, \varphi, Acc)$ such that:

- $R = (R_1 \times R_2) \cup R^{\tau}$;
- $R^{\tau} = \{\tau_{r_1,r_2} \mid r_1 \in R_1, r_2 \in R_2\} \cup \{\tau\}$;
- $r_0 = (r_{1,0}, r_{2,0})$;
- $(r_1, r_2) \Rightarrow (r_1', r_2')$ if, and only if, $r_1 \Rightarrow_1 r_1'$ and $r_2 \Rightarrow_2 r_2'$;
- $\forall r_1 \in R_1, r_2 \in R_2, (r_1, r_2) \Rightarrow \tau_{r_1,r_2}$ and $\tau_{r_1,r_2} \Rightarrow \tau$;
- $\varphi(r_1, r_2) = \varphi_1(r_1) \ \wedge \ \varphi_2(r_2)$;
- $\varphi(\tau_{r_1,r_2}) = \bigvee_{r_1 \Rightarrow_1 r_1'} \varphi_1(r_1') \ \wedge \bigvee_{\forall \alpha \in Acc(r_1,r_2), \sigma \notin \alpha} \sigma$;
- $\varphi(\tau) = True$;
- $Acc(r_1, r_2) = Acc_1(r_1)/Acc_2(r_2)$;
- $Acc(\tau_{r_1,r_2}) = Acc(\tau) = 2^{2^{2^{\Pi}}}$.

V. Conclusion

Interface Theory based on DPAA as specification, and which realizations are Kripke Models, has been defined and endowed with the composition operators of a complete interface theory, answering some needs for requirements engineering. Implicit representation of acceptance sets are transposable in the form of Boolean functions, defined by BDDs [15]. The next step will be the implementation of this algebra and its test on relevant case studies in system engineering.



	$Acc(r_i)$
r_0	$\{a\}, \{a, ab\}, \{a, ab, b\}, \{ab\}, \{ab, b\}, \{b\}, \emptyset$
r_1	$\{a\}, \{a, ab\}, \{ab\}$
r_2	$\{a\}$
r_3	$\{a\}, \emptyset$

Fig. 8. $\mathcal{A}_3 = \mathcal{A}_1 \otimes \mathcal{A}_2$ (parallel composition).

References

[1] B. Meyer, "Applying "design by contract"," *IEEE Computer*, vol. 25, no. 10, pp. 40–51, October 1992.

[2] R. B. Findler, M. Latendresse, and M. Felleisen, "Behavioral contracts and behavioral subtyping," in *ACM Conference Foundations of Software Engineering*, 2001.

[3] B. Meyer, *Touch of Class: Learning to Program Well Using Object Technology and Design by Contract*. Springer S. E., 2009.

[4] C. A. R. Hoare, "An axiomatic basis for computer programming," *Commun. ACM*, vol. 12, no. 10, pp. 576–583, 1969.

[5] P. W. O'Hearn, "Separation logic," *Commun. ACM*, vol. 62, no. 2, pp. 86–95, 2019.

[6] A. Benveniste, B. Caillaud, D. Nickovic, R. Passerone, J.-B. Raclet, P. Reinkemeier, A. Sangiovanni-Vincentelli, W. Damm, T. A. Henzinger, and K. G. Larsen, "Contracts for system design," *Foundations and Trends® in Electronic Design Automation*, vol. 12, no. 2-3, pp. 124–400, 2018.

[7] L. de Alfaro and T. A. Henzinger, "Interface automata," *SIGSOFT Softw. Eng. Notes*, vol. 26, no. 5, pp. 109–120, Sep. 2001.

[8] J. Raclet, E. Badouel, A. Benveniste, B. Caillaud, A. Legay, and R. Passerone, "A modal interface theory for component-based design," *Fundamenta Informaticae*, vol. 108, no. 1-2, pp. 119–149, 2011.

[9] F. Bujtor, S. Fendrich, G. Lüttgen, and W. Vogler, "Nondeterministic modal interfaces," in *SOFSEM 2015: Theory and Practice of Computer Science*. Springer Berlin Heidelberg, 2015, pp. 152–163.

[10] N. A. Lynch and E. W. Stark, "A proof of the kahn principle for input/output automata," *Inf. Comput.*, vol. 82, no. 1, pp. 81–92, 1989.

[11] N. A. Lynch, "Input/output automata: Basic, timed, hybrid, probabilistic, dynamic," in *CONCUR*, 2003, pp. 187–188.

[12] A. Chakrabarti, L. de Alfaro, T. A. Henzinger, and F. Y. C. Mang, "Synchronous and bidirectional component interfaces," in *International Conference on Computer Aided Verification*, ser. Lecture Notes in Computer Science, vol. 2404. Springer, 2002, pp. 414–427.

[13] J. Raclet, "Quotient de spécifications pour la réutilisation de composants," Ph.D. dissertation, Université de Rennes 1, 2007.

[14] L. Lamport, "What good is temporal logic?" *Information Processing, R. E. A. Mason, ed., Elsevier Publishers*, vol. 83, pp. 657–668, May 1983.

[15] Bryant, "Graph-based algorithms for boolean function manipulation," *IEEE Transactions on Computers*, vol. C-35, no. 8, pp. 677–691, Aug 1986.

[16] F. Moller and S. Smolka, "On the computational complexity of bisimulation," *ACM Comput. Surv.*, vol. 27, pp. 287–289, 06 1995.